ProTrain: Efficient LLM Training via Automatic Memory Management

Hanmei Yang, Jin Zhou, Hui Guan, Tongping Liu
University of Massachusetts Amherst

Yao Fu, Xiaoqun Wang, Ramine Roane
Advanced Micro Devices, Inc.

MLSys 2026

LLM Training: Memory is the Bottleneck

The Memory Wall

16 bytes per parameter for model states

(fp16 weights + fp16 grads + fp32 master weights + fp32 momentum + fp32 variance)

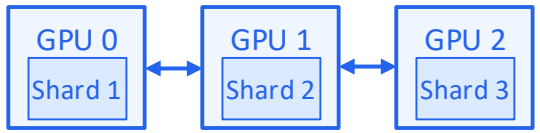
**Excluding Activations memory*

Example: LLaMA-2 70B

70B × 16 bytes = 1.12 TB

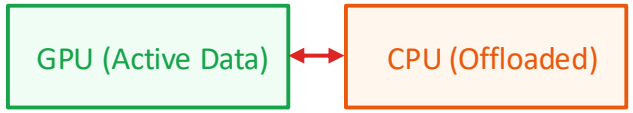
Requires 14× more memory than a single 80GB A100 GPU

Memory Optimization Techniques



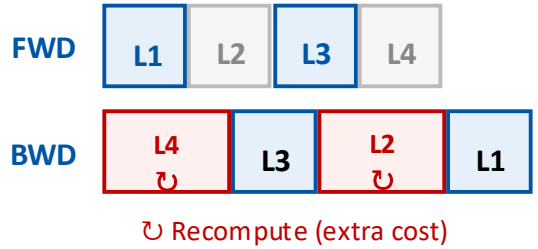
ZeRO / FSDP

Partitions model states across GPUs to reduce redundancy



Offloading / Swapping

Leverages CPU memory to extend GPU capacity



Gradient Checkpointing

Trades compute for memory by recomputing activations

These Techniques Conflict and Compete

Technique Selection

MODEL STATES

ZeRO Sharding vs. Param. Offloading

Keep in full on GPU, or partition and offload to CPU?

ACTIVATION MEMORY

Grad. Checkpointing vs. Act. Swapping

Recompute or offload to CPU?

Resource Contention

PCIE BANDWIDTH

Act. Swapping vs. Param. Offloading

Both need PCIe — saturating one starves the other

GPU MEMORY

Resident Param. vs. Prefetch Buffers

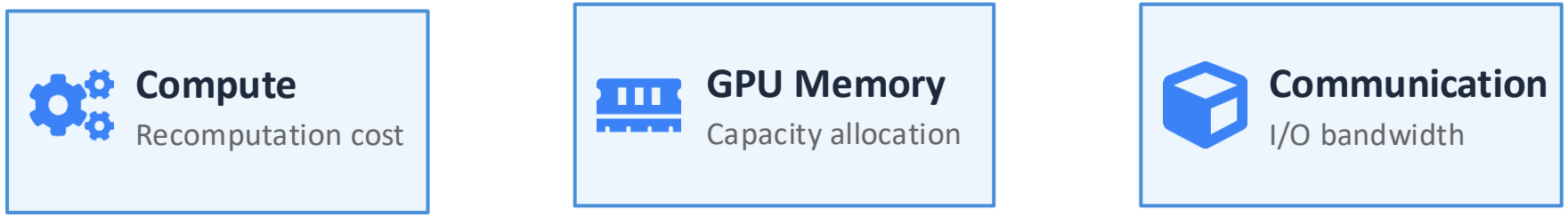
Keeping more params resident leaves less room for prefetching

Existing frameworks lack effective coordination of these techniques

e.g., DeepSpeed: 18+ low-level knobs, many coupled → default: 35.6% mem used, 1.18× slower

The Real Problem: Coordination

Key Insight: Memory optimization for LLM training is a ***joint coordination problem*** across compute, memory, and communication — not a collection of independent techniques.



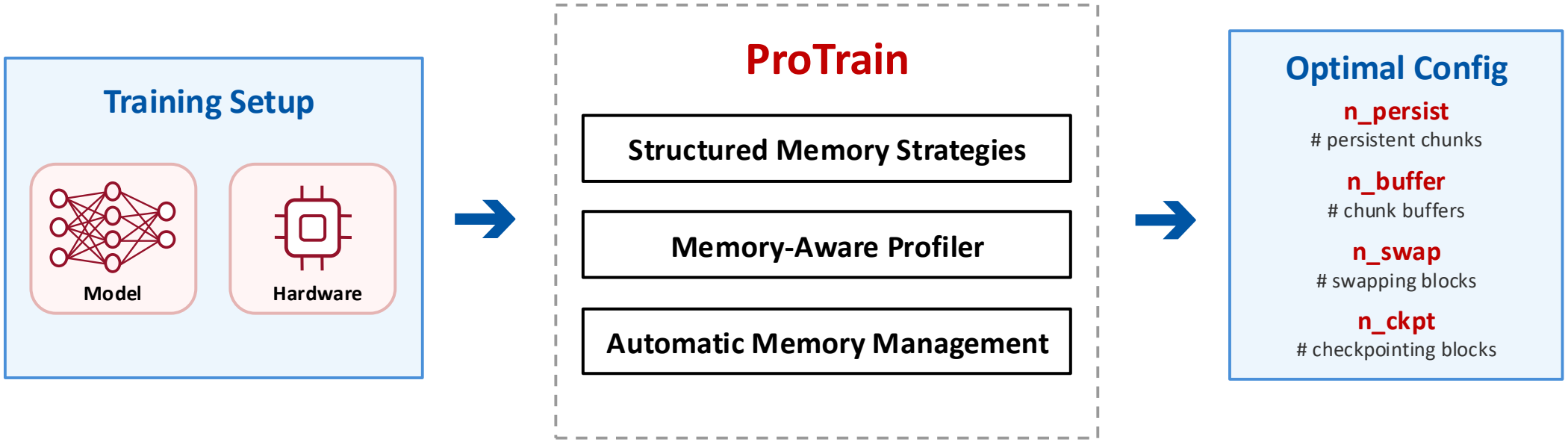
Key Challenges:

1. **Lack of Clean Abstractions** → **Structured Memory Strategies**
   Existing frameworks expose an unstructured parameter space with complex interdependencies

2. **Inaccurate Memory Estimation** → **Memory-Aware Profiler**
   Traditional profilers underestimate actual memory usage

3. **Manual Tuning Doesn't Scale** → **Automatic Memory Management**
   Search space too large for manual exploration; optimal configuration varies across hardware

ProTrain: System Overview



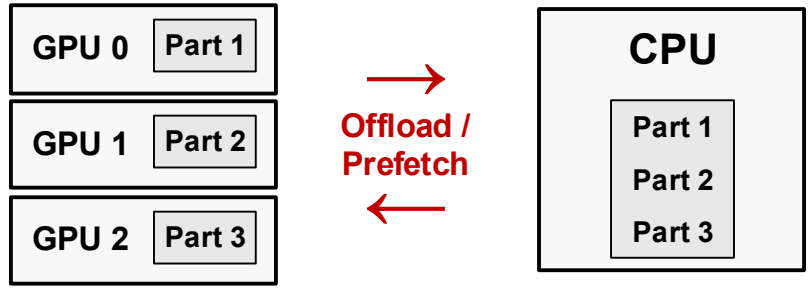
Optimization Objective

minimize T(n_persist, n_buffer, n_swap, n_ckpt) per-iteration time

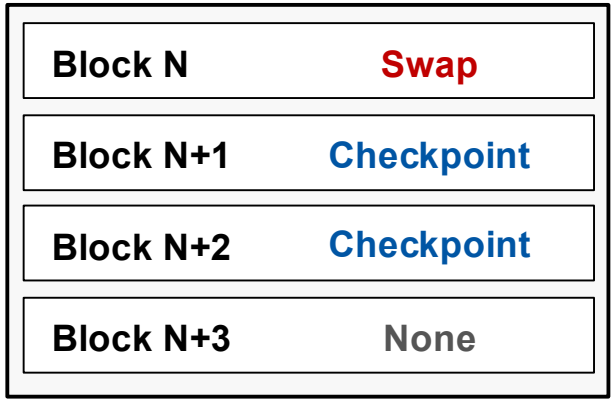
subject to M(n_persist, n_buffer, n_swap, n_ckpt) $\leq C$ GPU memory capacity

Structured Memory Strategies

- **Model States** → **Hierarchical Chunk Management**
  - Combines ZeRO Sharding and CPU Offloading
  - Organizes model states (i.e., parameters, gradients, and optimizer states) into uniform chunks for efficient I/O transfers



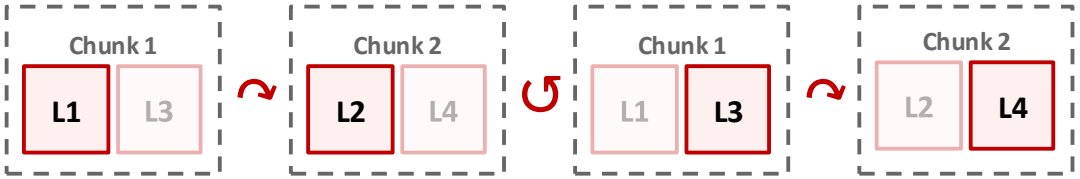
- **Activations** → **Interleaved Block Management**
  - Combines Gradient Checkpointing and Activation Swapping
  - Assigns each transformer block an independent activation strategy (Checkpointing / Swapping / None)



Hierarchical Chunk Management

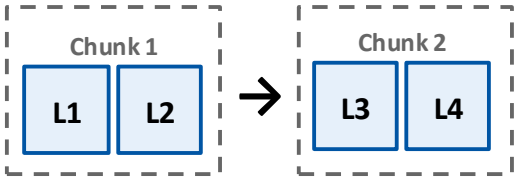
Intra-Chunk Level

Model-defined order:



Ping-pong access, cache thrashing

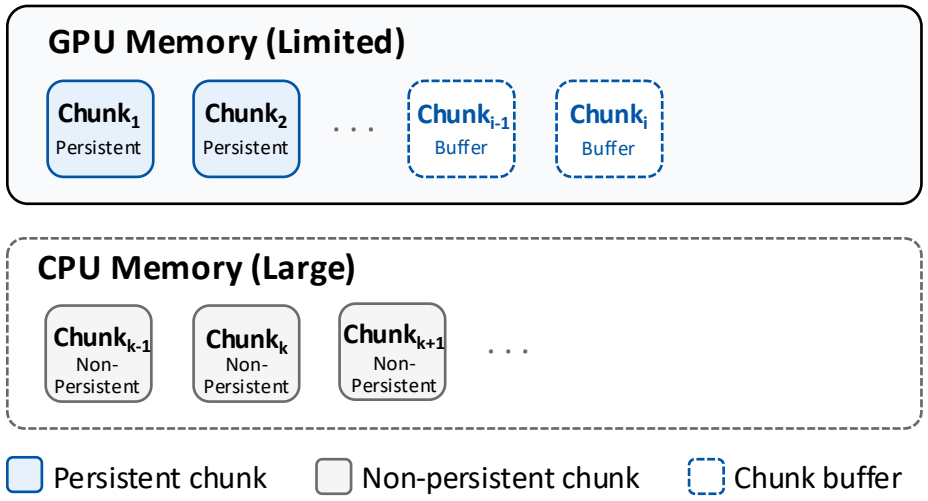
Execution order:



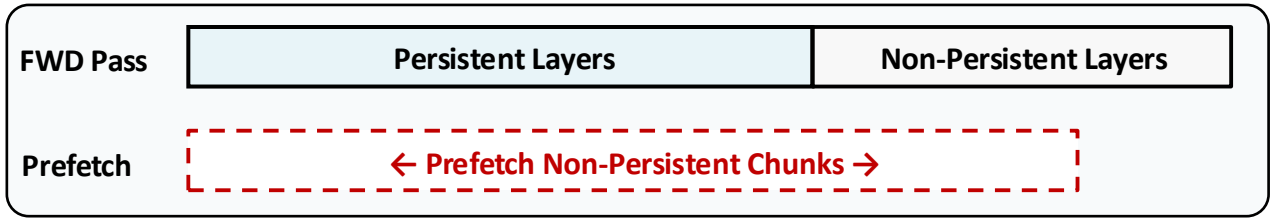
Sequential access, early release

* Adapts to reversed access order when using gradient checkpointing

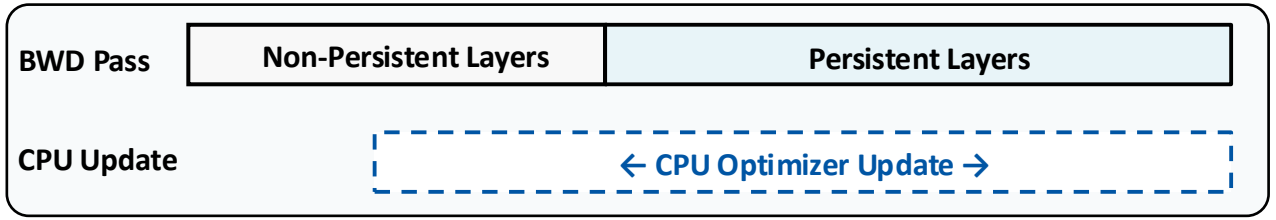
Inter-Chunk Level



FWD: Persistent chunks for early layers → longer prefetch window



BWD: CPU optimizer update overlaps with GPU backward computation

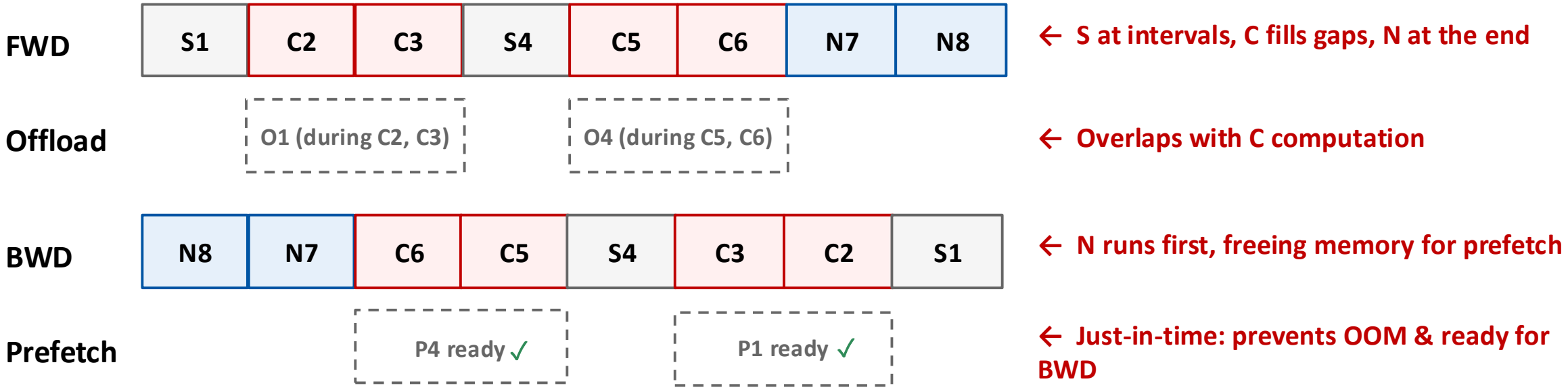


Interleaved Block Management

- **Block-wise Activation Management**

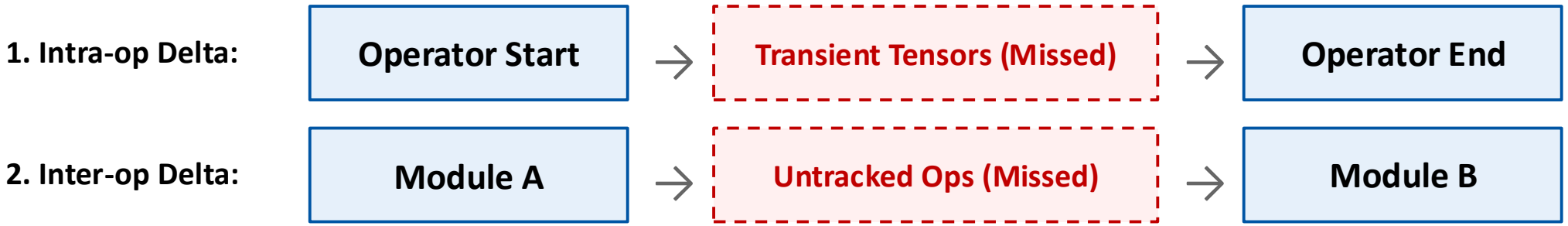
S	**Swap:** Offload to CPU	C	**Checkpoint:** Recompute	N	**None:** Keep on GPU

- **Interleaved Layout**



Memory-Aware Profiler

Why Traditional Profiling Fails

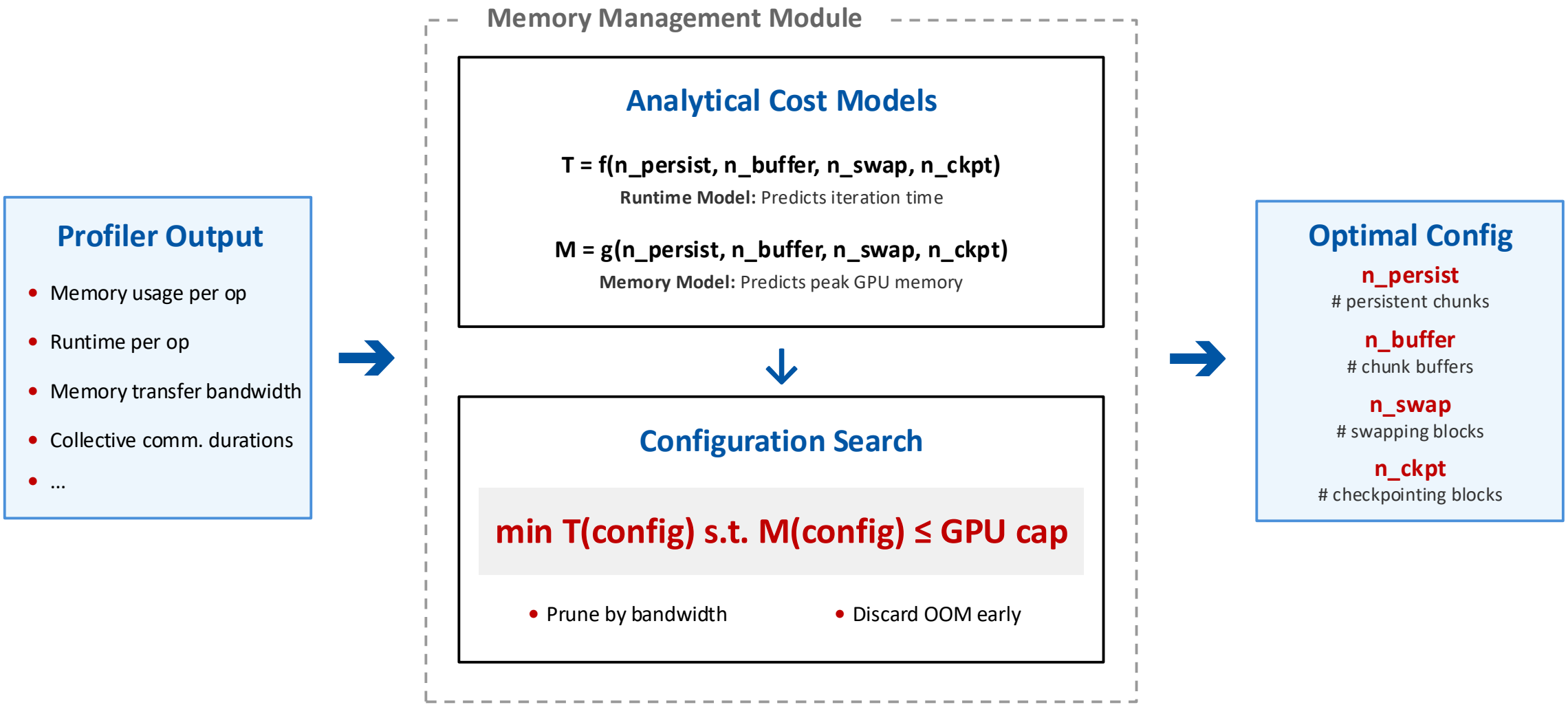


→ **Missed 17.2% (3.06 GB) of peak memory in 10B GPT-2**

Memory-Aware Profiling

- **Full Model Execution Trace**: Profile all operators in a single complete forward pass.
- **Drop Model States & Activations**: On-demand tensor management reduces memory to fit single GPU.
- **Measure Both Memory Deltas**: Capture both intra-op (transient) and inter-op (unhookable) memory.
- **Reconstruct Actual Peak Memory**: Combine static analysis with operator-level tracking for accurate estimates.

Automatic Memory Management



Evaluation Setup

- **Baselines**
  - DeepSpeed
  - Colossal-AI
  - PyTorch FSDP
- **Models**
  - Mistral 7B
  - OPT 13B, 30B
  - LlaMA-2 13B, 34B
  - GPT-2 10B, 15B, 20B, 30B, 40B

Hardware

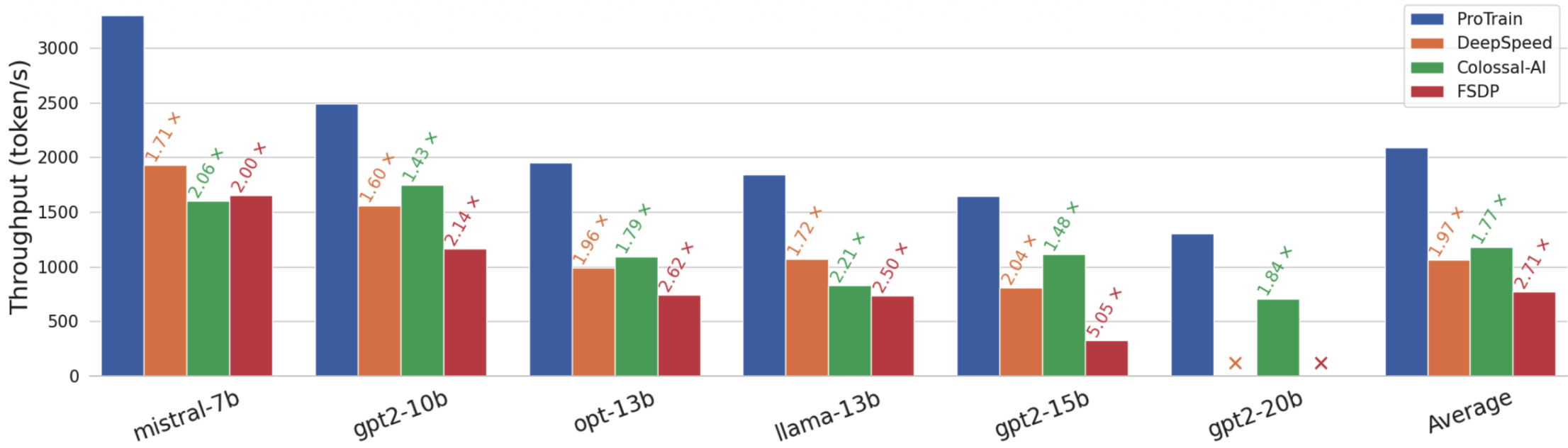
	Server 1	Server 2
GPU	4× RTX 3090 (24GB)	4× A100 (80GB)
CPU	Xeon(R) Silver 4214R (384GB)	Xeon(R) Platinum 8480+ (1TB)
CPU-GPU Interconnect	PCIe 3.0 (15.8 GB/s)	PCIe 4.0 (31.5GB/s)
GPU-GPU Interconnect	PCIe 3.0 (15.8 GB/s)	NVLink 3.0 (300GB/s)

Maximum Trainable Model Size

Hardware Setup	ProTrain	DeepSpeed	Colossal-AI	FSDP
1× RTX 3090	**34B**	15B	25B	3B
4× RTX 3090	**37B**	15B	25B	15B
1× A100	**75B**	34B	53B	10B
4× A100	**87B**	37B	53B	55B

ProTrain scales up to **87B** on 4×A100 — **2.35×** larger than DeepSpeed, **1.64×** larger than Colossal-AI, and **1.58×** larger than FSDP.

Training Throughput (4× RTX 3090)



ProTrain achieves the highest throughput across all models – on average **1.97×** over DeepSpeed, **1.77×** over Colossal-AI, and **2.71×** over FSDP.

Conclusion

- **Problem**: Existing frameworks require extensive manual tuning of numerous memory-related parameters that are tightly coupled, and their system implementations lack unified optimization.
- **Solution**: ProTrain abstracts memory strategies into a structured 4-parameter space, with a memory-aware profiler and cost models that enable automatic configuration search.
- **Results**: Trains up to 87B parameters on 4×A100 (1.58–2.35× larger than baselines). Achieves 1.43–2.71× higher throughput across all models and hardware configurations.